A Different Type of Traversal

Strategy:

Traversals:

One Algorithm, Three Traversals:

<table>
<thead>
<tr>
<th>BinaryTree.cpp</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
<tr>
<td>51</td>
</tr>
<tr>
<td>52</td>
</tr>
<tr>
<td>53</td>
</tr>
<tr>
<td>54</td>
</tr>
<tr>
<td>55</td>
</tr>
<tr>
<td>56</td>
</tr>
<tr>
<td>57</td>
</tr>
</tbody>
</table>

Traversal vs. Search:
- **Traversal** visits every node in the tree exactly once.
- **Search** finds one (or more) element(s) in the tree.

Breadth First Traversal + Search:

Depth First Traversal + Search
Runtime Analysis on a Binary Tree:

- Find an element: Best case? Worst case?
- Insertion of a sorted list of elements: Best case? Worst case?
- Running time bound by

Dictionary ADT

Dictionary.h

```cpp
3
4 class Dictionary {
5   public:
6     ...
7 ...
13   private:
14   ...
16   }
```

A Searchable Binary Tree?

Binary Search Tree Property:

Finding an element in a BST:

BST.cpp

```cpp
template <typename K, typename V>
______________ find(const K & key) {

}

template <typename K, typename V>
______________ _find

(TreeNode * & root, const K & key) {

}
```

CS 225 – Things To Be Doing:

1. mp_list due Sunday.
2. lab_inheritance starts today
3. exam 1 reschedule window Saturday 2/26 – Monday 2/28.
4. Daily POTDs